CLAIMS

What is claimed is:

1. (currently amended) A method in a data processing system for developing source code comprising the steps of:

generating a transient meta model which stores a language-neutral representation of the source code;

displaying a graphical representation of the source code generated from the language-neutral representation in the transient meta model;

receiving a message corresponding to a portion of the source code;

locating the portion of the source code corresponding to the message;

displaying the portion of the source code corresponding to the message in a visually distinctive manner;

determining whether the graphical representation of the portion of the source code corresponding to the message is displayed; and

when it is determined that the graphical representation of the portion of the source code corresponding to the message is not displayed, displaying the graphical representation of the portion of the source code corresponding to the message; and

modifying the graphical representation of the portion of the source code corresponding to the message in a visually distinctive manner.

2. (original) The method of claim 1, wherein the message is received from a verification tool.

3. (original) The method of claim 2, wherein the verification tool comprises a compiler.

4. (original) The method of claim 2, wherein the verification tool comprises a quality assurance module.

5. (original) The method of claim 1, wherein the message comprises an error message.

6. (original) The method of claim 1, wherein the message comprises a line number of the source code.

7. (original) The method of claim 6, wherein the portion of the source code corresponding to the message is located using the line number.

8. (original) The method of claim 1, wherein the message comprises a name of a file containing the source code.

9. (original) The method of claim 1, wherein the graphical representation comprises a class diagram.

10. (currently amended) A method in a data processing system for developing source code comprising the steps of:

generating a transient meta model which stores a language-neutral representation of the source code;

displaying a graphical representation of the source code generated from the language-neutral representation in the transient meta model ;

receiving a message corresponding to a portion of the source code;

locating the portion of the source code corresponding to the message;

displaying the portion of the source code corresponding to the message; and

displaying the graphical representation of the portion of the source code corresponding to the message in a visually distinctive manner.

11. (original) The method of claim 10, further comprising the steps of:

detecting an error in the source code; and

generating a message reflecting the error.

12. (original) The method of claim 10, wherein the message is received from a verification tool.

13. (original) The method of claim 12, wherein the verification tool comprises a compiler.

14. (original) The method of claim 12, wherein the verification tool comprises a quality assurance module.

15. (original) The method of claim 10, wherein the message comprises an error message.

16. (original) The method of claim 10, wherein the message comprises a line number of the source code.

17. (original) The method of claim 16, wherein the portion of the source code corresponding to the message is located using the line number.

18. (original) The method of claim 10, wherein the message comprises a name of a file containing the source code.

19. (original) The method of claim 10, wherein the graphical representation comprises a class diagram.

20. (currently amended) A method in a data processing system for developing source code comprising the steps of:

generating a transient meta model which stores a language-neutral representation of the source code;

receiving a message corresponding to a portion of the source code; and

displaying the graphical representation generated from the language-neutral representation in the transient meta model pertaining to [[of]] the portion of the source code corresponding to the message in a visually distinctive manner.

21. (original) The method of claim 20, wherein the message is received from a verification tool.

22. (original) The method of claim 21, wherein the verification tool comprises a compiler.

23. (original) The method of claim 21, wherein the verification tool comprises a quality assurance module.

24. (original) The method of claim 20, wherein the message comprises an error message.

25. (original) The method of claim 20, wherein the message comprises a line number of the source code.

26. (original) The method of claim 25, wherein the portion of the source code corresponding to the message is located using the line number.

27. (original) The method of claim 20, wherein the message comprises a name of a file containing the source code.

28. (original) The method of claim 20, wherein the graphical representation comprises a class diagram.

29. (original) The method of claim 20, wherein the message comprises a basic metrics message.

30. (original) The method of claim 29, wherein the basic metrics message comprises a lines of code message.

31. (original) The method of claim 29, wherein the basic metrics message comprises a number of attributes message.

32. (original) The method of claim 29, wherein the basic metrics message comprises a number of classes message.

33. (original) The method of claim 29, wherein the basic metrics message comprises a number of constructors message.

34. (original) The method of claim 29, wherein the basic metrics message comprises a number of import statements message.

35. (original) The method of claim 29, wherein the basic metrics message comprises a number of members message.

36. (original) The method of claim 29, wherein the basic metrics message comprises a number of operations message.

37. (original) The method of claim 20, wherein the message comprises a cohesion metrics message.

38. (original) The method of claim 37, wherein the cohesion metrics message comprises a lack of cohesion message.

39. (original) The method of claim 20, wherein the message comprises a complexity metrics message.

40. (original) The method of claim 39, wherein the complexity metrics message comprises an attribute complexity message.

41. (original) The method of claim 39, wherein the complexity metrics message comprises a cyclomatic complexity message.

42. (original) The method of claim 39, wherein the complexity metrics message comprises a number of remote methods message.

43. (original) The method of claim 39, wherein the complexity metrics message comprises a response for class message.

44. (original) The method of claim 39, wherein the complexity metrics message comprises a weighted methods per class message.

45. (original) The method of claim 20, wherein the message comprises a coupling metrics message.

46. (original) The method of claim 45, wherein the coupling metrics message comprises a coupling between objects message.

47. (original) The method of claim 45 wherein the coupling metrics message comprises a coupling factor message.

48. (original) The method of claim 45, wherein the coupling metrics message comprises a data abstraction coupling message.

49. (original) The method of claim 45, wherein the coupling metrics message comprises a fanout message.

50. (original) The method of claim 20, wherein the message comprises a Halstead metrics message.

51. (original) The method of claim 50, wherein the Halstead metrics message comprises a Halstead difficulty message.

52. (original) The method of claim 50, wherein the Halstead metrics message comprises a Halstead effort message.

53. (original) The method of claim 50, wherein the Halstead metrics message comprises a Halstead program length message.

54. (original) The method of claim 50, wherein the Halstead metrics message comprises a Halstead program vocabulary message.

55. (original) The method of claim 50, wherein the Halstead metrics message comprises a Halstead program volume message.

56. (original) The method of claim 50, wherein the Halstead metrics message comprises a number of operands message.

57. (original) The method of claim 50, wherein the Halstead metrics message comprises a number of operators message.

58. (original) The method of claim 50, wherein the Halstead metrics message comprises a number of unique operands message.

59. (original) The method of claim 50, wherein the Halstead metrics message comprises a number of unique operators message.

60. (original) The method of claim 20, wherein the message comprises an incapsulation metrics message.

61. (original) The method of claim 60, wherein the incapsulation metrics message comprises an attribute hiding factor message.

62. (original) The method of claim 60, wherein the incapsulation metrics message comprises a method hiding factor message.

63. (original) The method of claim 20, wherein the message comprises an inheritance metrics message.

64. (original) The method of claim 63, wherein the inheritance metrics message comprises an attribute inheritance factor message.

65. (original) The method of claim 63, wherein the inheritance metrics message comprises a depth of inheritance hierarchy message.

66. (original) The method of claim 63, wherein the inheritance metrics message comprises a method inheritance factor message.

67. (original) The method of claim 63, wherein the inheritance metrics message comprises a number of child classes message.

68. (original) The method of claim 20, wherein the message comprises a maximum metrics message.

69. (original) The method of claim 68, wherein the maximum metrics message comprises a maximum number of levels message.

70. (original) The method of claim 68, wherein the maximum metrics message comprises a maximum number of parameters message.

71. (original) The method of claim 68, wherein the maximum metrics message comprises a maximum size of operation message.

72. (original) The method of claim 20, wherein the message comprises a polymorphism metrics message.

73. (original) The method of claim 72, wherein the polymorphism metrics message comprises a number of added methods message.

74. (original) The method of claim 72, wherein the polymorphism metrics message comprises a number of overridden methods message.

75. (original) The method of claim 72, wherein the polymorphism metrics message comprises a polymorphism factor message.

76. (original) The method of claim 20, wherein the message comprises a ratio metrics message.

77. (original) The method of claim 76, wherein the ratio metrics message comprises a comment ratio message.

78. (original) The method of claim 76, wherein the ratio metrics message comprises a percentage of package members message.

79. (original) The method of claim 76, wherein the ratio metrics message comprises a percentage of private members message.

80. (original) The method of claim 76, wherein the ratio metrics message comprises a percentage of protected members message.

81. (original) The method of claim 76, wherein the ratio metrics message comprises a percentage of public members message.

82. (original) The method of claim 76, wherein the ratio metrics message comprises a true comment ratio message.

83. (original) The method of claim 20, wherein the message comprises a coding style audits message.

84. (original) The method of claim 83, wherein the coding style audits message comprises an avoid complex initialization or update clause in for loops message.

85. (original) The method of claim 83, wherein the coding style audits message comprises an avoid implementation packages referencing message.

86. (original) The method of claim 83, wherein the coding style audits message comprises an access of static members through objects message.

87. (original) The method of claim 83, wherein the coding style audits message comprises an assignment to formal parameters message.

88. (original) The method of claim 83, wherein the coding style audits message comprises an avoid too long files message.

89. (original) The method of claim 83, wherein the coding style audits message comprises an avoid too long lines message.

90. (original) The method of claim 83, wherein the coding style audits message comprises a complex assignment message.

91. (original) The method of claim 83, wherein the coding style audits message comprises a don't code numerical constants directly message.

92. (original) The method of claim 83, wherein the coding style audits message comprises a don't place multiple statements on the same line message.

93. (original) The method of claim 83, wherein the coding style audits message comprises a don't use the negation operator frequently message.

94. (original) The method of claim 83, wherein the coding style audits message comprises an operator '?:' may not be used message.

95. (original) The method of claim 83, wherein the coding style audits message comprises a parenthesize conditional part of ternary conditional expression message.

96. (original) The method of claim 83, wherein the coding style audits message comprises a put declarations only at the beginning of blocks message.

97. (original) The method of claim 83, wherein the coding style audits message comprises a provide incremental in for-statement or use while-statement message.

98. (original) The method of claim 83, wherein the coding style audits message comprises a replacement for demand imports message.

99. (original) The method of claim 83, wherein the coding style audits message comprises a switch statement should include a default case message.

100. (original) The method of claim 83, wherein the coding style audits message comprises a use abbreviated assignment operator message.

101. (original) The method of claim 83, wherein the coding style audits message comprises a use 'this' explicitly to access class members message.

102. (original) The method of claim 20, wherein the message comprises a critical errors audits message.

103. (original) The method of claim 102, wherein the critical errors audits message comprises an avoid hiding inherited attributes message.

104. (original) The method of claim 102, wherein the critical errors audits message comprises an avoid hiding inherited static methods message.

105. (original) The method of claim 102, wherein the critical errors audits message comprises a command query separation message.

106. (original) The method of claim 102, wherein the critical errors audits message comprises a hiding of names message.

107. (original) The method of claim 102, wherein the critical errors audits message comprises an inaccessible constructor or method matches message.

108. (original) The method of claim 102, wherein the critical errors audits message comprises a multiple visible declarations with same name message.

109. (original) The method of claim 102, wherein the critical errors audits message comprises an overriding a non-abstract method with an abstract method message.

110. (original) The method of claim 102, wherein the critical errors audits message comprises an overriding a private method message.

111. (original) The method of claim 102, wherein the critical errors audits message comprises an overloading within a subclass message.

112. (original) The method of claim 102, wherein the critical errors audits message comprises a use of static attribute for initialization message.

113. (original) The method of claim 20, wherein the message comprises a declaration style audits message.

114. (original) The method of claim 113, wherein the declaration style audits message comprises a badly located array declarators message.

115. (original) The method of claim 113, wherein the declaration style audits message comprises a constant private attributes must be final message.

116. (original) The method of claim 113, wherein the declaration style audits message comprises a constant variables must be final message.

117. (original) The method of claim 113, wherein the declaration style audits message comprises a declare variables in one statement each message.

118. (original) The method of claim 113, wherein the declaration style audits message comprises an instantiated classes should be final message.

119. (original) The method of claim 113, wherein the declaration style audits message comprises a list all public and package members first message.

120. (original) The method of claim 113, wherein the declaration style audits message comprises an order of class members declaration message.

121. (original) The method of claim 113, wherein the declaration style audits message comprises an order of appearance of modifiers message.

122. (original) The method of claim 113, wherein the declaration style audits message comprises a put the main function last message.

123. (original) The method of claim 113, wherein the declaration style audits message comprises a place public class first message.

124. (original) The method of claim 20, wherein the message comprises a documentation audits message.

125. (original) The method of claim 124, wherein the documentation audits message comprises a bad tag in JavaDoc comments message.

126. (original) The method of claim 124, wherein the documentation audits message comprises a distinguish between JavaDoc and ordinary comments message.

127. (original) The method of claim 124, wherein the documentation audits message comprises a provide file comments message.

128. (original) The method of claim 124, wherein the documentation audits message comprises a provide JavaDoc comments message.

129. (original) The method of claim 20, wherein the message comprises a naming style audits message.

130. (original) The method of claim 129, wherein the naming style audits message comprises a class name must match its file name message.

131. (original) The method of claim 129, wherein the naming style audits message comprises a group operations with same name together message.

132. (original) The method of claim 129, wherein the naming style audits message comprises a naming conventions message.

133. (original) The method of claim 129, wherein the naming style audits message comprises a names of exception classes message.

134. (original) The method of claim 129, wherein the naming style audits message comprises a use conventional variable names message.

135. (original) The method of claim 20, wherein the message comprises a performance audits message.

136. (original) The method of claim 135, wherein the performance audits message comprises an avoid declaring variables inside loops message.

137. (original) The method of claim 135, wherein the performance audits message comprises an append to string within a loop message.

138. (original) The method of claim 135, wherein the performance audits message comprises a complex loop expressions message.

139. (original) The method of claim 20, wherein the message comprises a possible error audits message.

140. (original) The method of claim 139, wherein the possible error audits message comprises an avoid empty catch blocks message.

141. (original) The method of claim 139, wherein the possible error audits message comprises an avoid public and package attributes message.

142. (original) The method of claim 139, wherein the possible error audits message comprises an avoid statements with empty body message.

143. (original) The method of claim 139, wherein the possible error audits message comprises an assignment to for-loop variables message.

144. (original) The method of claim 139, wherein the possible error audits message comprises a don't compare floating point types message.

145. (original) The method of claim 139, wherein the possible error audits message comprises an enclosing body within a block message.

146. (original) The method of claim 139, wherein the possible error audits message comprises an explicitly initialize all variables message.

147. (original) The method of claim 139, wherein the possible error audits message comprises a method finalize() doesn't call super.finalize() message.

148. (original) The method of claim 139, wherein the possible error audits message comprises a mixing logical operators without parentheses message.

149. (original) The method of claim 139, wherein the possible error audits message comprises a no assignments in conditional expressions message.

150. (original) The method of claim 139, wherein the possible error audits message comprises a supply break or comment in case statement message.

151. (original) The method of claim 139, wherein the possible error audits message comprises a use 'equals' instead of '==' message.

152. (original) The method of claim 139, wherein the possible error audits message comprises a use 'L' instead of 'l' at the end of integer constant message.

153. (original) The method of claim 139, wherein the possible error audits message comprises a use of the 'synchronized' modifier message.

154. (original) The method of claim 20, wherein the message comprises a superfluous content audits message.

155. (original) The method of claim 154, wherein the superfluous content audits message comprises a duplicate import declarations message.

156. (original) The method of claim 154, wherein the superfluous content audits message comprises a don't import the package the source file belongs to message.

157. (original) The method of claim 154, wherein the superfluous content audits message comprises an explicit import of the java.lang classes message.

158. (original) The method of claim 154, wherein the superfluous content audits message comprises an equality operations on boolean arguments message.

159. (original) The method of claim 154, wherein the superfluous content audits message comprises an imported items must be used message.

160. (original) The method of claim 154, wherein the superfluous content audits message comprises an unnecessary casts message.

161. (original) The method of claim 154, wherein the superfluous content audits message comprises an unnecessary 'instanceof' evaluations message.

162. (original) The method of claim 154, wherein the superfluous content audits message comprises an unused local variables and formal parameters message.

163. (original) The method of claim 154, wherein the superfluous content audits message comprises a use of obsolete interface modifier message.

164. (original) The method of claim 154, wherein the superfluous content audits message comprises a use of unnecessary interface member modifiers message.

165. (original) The method of claim 154, wherein the superfluous content audits message comprises an unused private class member message.

166. (original) The method of claim 154, wherein the superfluous content audits message comprises an unnecessary return statement parentheses message.

167. (currently amended) A method in a data processing system for developing source code comprising the steps of:

generating a transient meta model which stores a language-neutral representation of the source code;

displaying a graphical representation of the source code generated from the language-neutral representation in the transient meta model;

detecting an error in the source code;

generating a message reflecting the error;

locating a portion of the source code corresponding to the message;

displaying the portion of the source code corresponding to the message in a visually distinctive manner;

determining whether the graphical representation of the portion of the source code corresponding to the message is displayed; and

when it is determined that the graphical representation of the portion of the source code corresponding to the message is not displayed, displaying the graphical representation of the portion of the source code corresponding to the message; and

modifying the graphical representation of the portion of the source code corresponding to the message in a visually distinctive manner.

168. (original) The method of claim 167, wherein the message is generated by a verification tool.

169. (original) The method of claim 168, wherein the verification tool comprises a compiler.

170. (original) The method of claim 168, wherein the verification tool comprises a quality assurance module.

171. (original) The method of claim 167, wherein the message comprises an error message.

172. (original) The method of claim 167, wherein the message comprises a line number of the source code.

173. (original) The method of claim 172, wherein the portion of the source code corresponding to the message is located using the line number.

174. (original) The method of claim 167, wherein the message comprises a name of a file containing the source code.

175. (original) The method of claim 167, wherein the graphical representation comprises a class diagram.

176. (currently amended) A method in a data processing system for developing source code comprising the steps of:

generating a transient meta model which stores a language-neutral representation of the source code;

displaying a graphical representation of the source code generated from the language-neutral representation in the transient meta model;

detecting an error in the source code;

generating a message reflecting the error;

locating a portion of the source code corresponding to the message;

displaying the portion of the source code corresponding to the message; and

displaying the graphical representation of the portion of the source code corresponding to the message in a visually distinctive manner.

177. (original) The method of claim 176, wherein the message is generated by a verification tool.

178. (original) The method of claim 177, wherein the verification tool comprises a compiler.

179. (original) The method of claim 177, wherein the verification tool comprises a quality assurance module.

180. (original) The method of claim 176, wherein the message comprises an error message.

181. (original) The method of claim 176, wherein the message comprises a line number of the source code.

182. (original) The method of claim 181, wherein the portion of the source code corresponding to the message is located using the line number.

183. (original) The method of claim 176, wherein the message comprises a name of a file containing the source code.

184. (original) The method of claim 176, wherein the graphical representation comprises a class diagram.

185. (currently amended) A method in a data processing system for developing source code comprising the steps of:

generating a transient meta model which stores a language-neutral representation of the source code;

detecting an error in the source code;

generating a message reflecting the error; and

displaying the graphical representation generated from the language-neutral representation in the transient meta model pertaining to [[of]] a portion of the source code corresponding to the message in a visually distinctive manner.

186. (original) The method of claim 185, wherein the message is generated by a verification tool.

187. (original) The method of claim 186, wherein the verification tool comprises a compiler.

188. (original) The method of claim 186, wherein the verification tool comprises a quality assurance module.

189. (original) The method of claim 185, wherein the message comprises an error message.

190. (original) The method of claim 185, wherein the message comprises a line number of the source code.

191. (original) The method of claim 190, wherein the portion of the source code corresponding to the message is located using the line number.

192. (original) The method of claim 185, wherein the message comprises a name of a file containing the source code.

193. (original) The method of claim 185, wherein the graphical representation comprises a class diagram.

194. (currently amended) A computer-readable medium containing instructions for controlling a data processing system to perform a method, the data processing system having source code, the method comprising the steps of:

generating a transient meta model which stores a language-neutral representation of the source code;

displaying a graphical representation of the source code generated from the language-neutral representation in the transient meta model;

receiving a message corresponding to a portion of the source code;

locating the portion of the source code corresponding to the message;

displaying the portion of the source code corresponding to the message in a visually distinctive manner;

determining whether the graphical representation of the portion of the source code corresponding to the message is displayed; and

when it is determined that the graphical representation of the portion of the source code corresponding to the message is not displayed, displaying the graphical representation of the portion of the source code corresponding to the message; and

modifying the graphical representation of the portion of the source code corresponding to the message in a visually distinctive manner.

195. (original) The computer-readable medium of claim 194, wherein the message is received from a verification tool.

196. (original) The computer-readable medium of claim 195, wherein the verification tool comprises a compiler.

197. (original) The computer-readable medium of claim 195, wherein the verification tool comprises a quality assurance module.

198. (original) The computer-readable medium of claim 194, wherein the message comprises an error message.

199. (original) The computer-readable medium of claim 194, wherein the message comprises a line number of the source code.

200. (original) The computer-readable medium of claim 199, wherein the portion of the source code corresponding to the message is located using the line number.

201. (original) The computer-readable medium of claim 194, wherein the message comprises a name of a file containing the source code.

202. (original) The computer-readable medium of claim 194, wherein the graphical representation comprises a class diagram.

203. (currently amended) A computer-readable medium containing instructions for controlling a data processing system to perform a method, the data processing system having source code, the method comprising the steps of:

generating a transient meta model which stores a language-neutral representation of the source code;

displaying a graphical representation of the source code generated from the language-neutral representation in the transient meta model;

receiving a message corresponding to a portion of the source code;

locating the portion of the source code corresponding to the message;

displaying the portion of the source code corresponding to the message; and

displaying the graphical representation of the portion of the source code corresponding to the message in a visually distinctive manner.

204. (original) The computer-readable medium of claim 203, wherein the method further comprises the steps of:

detecting an error in the source code; and

generating a message reflecting the error.

205. (original) The computer-readable medium of claim 203, wherein the message is received from a verification tool.

206. (original) The computer-readable medium of claim 205, wherein the verification tool comprises a compiler.

207. (original) The computer-readable medium of claim 205, wherein the verification tool comprises a quality assurance module.

208. (original) The computer-readable medium of claim 203, wherein the message comprises an error message.

209. (original) The computer-readable medium of claim 203, wherein the message comprises a line number of the source code.

210. (original) The computer-readable medium of claim 209, wherein the portion of the source code corresponding to the message is located using the line number.

211. (original) The computer-readable medium of claim 203, wherein the message comprises a name of a file containing the source code.

212. (original) The computer-readable medium of claim 203, wherein the graphical representation comprises a class diagram.

213. (currently amended) A computer-readable medium containing instructions for controlling a data processing system to perform a method, the data processing system having source code, the method comprising the steps of:

generating a transient meta model which stores a language-neutral representation of the source code;

receiving a message corresponding to a portion of the source code; and

displaying the graphical representation generated from the language-neutral representation in the transient meta model pertaining to [[of]] the portion of the source code corresponding to the message in a visually distinctive manner.

214. (original) The computer-readable medium of claim 213, wherein the message is received from a verification tool.

215. (original) The computer-readable medium of claim 214, wherein the verification tool comprises a compiler.

216. (original) The computer-readable medium of claim 214, wherein the verification tool comprises a quality assurance module.

217. (original) The computer-readable medium of claim 213, wherein the message comprises an error message.

218. (original) The computer-readable medium of claim 213, wherein the message comprises a line number of the source code.

219. (original) The computer-readable medium of claim 218, wherein the portion of the source code corresponding to the message is located using the line number.

220. (original) The computer-readable medium of claim 213, wherein the message comprises a name of a file containing the source code.

221. (original) The computer-readable medium of claim 213, wherein the graphical representation comprises a class diagram.

222. (currently amended) A computer-readable medium containing instructions for controlling a data processing system to perform a method, the data processing system having source code, the method comprising the steps of:

generating a transient meta model which stores a language-neutral representation of the source code;

displaying a graphical representation of the source code generated from the language-neutral representation in the transient meta model;

detecting an error in the source code;

generating a message reflecting the error;

locating a portion of the source code corresponding to the message;

displaying the portion of the source code corresponding to the message in a visually distinctive manner;

determining whether the graphical representation of the portion of the source code corresponding to the message is displayed; and

when it is determined that the graphical representation of the portion of the source code corresponding to the message is not displayed, displaying the graphical representation of the portion of the source code corresponding to the message; and

modifying the graphical representation of the portion of the source code corresponding to the message in a visually distinctive manner.

223. (original) The computer-readable medium of claim 222, wherein the message is generated by a verification tool.

224. (original) The computer-readable medium of claim 223, wherein the verification tool comprises a compiler.

225. (original) The computer-readable medium of claim 223, wherein the verification tool comprises a quality assurance module.

226. (original) The computer-readable medium of claim 222, wherein the message comprises an error message.

227. (original) The computer-readable medium of claim 222, wherein the message comprises a line number of the source code.

228. (original) The computer-readable medium of claim 227, wherein the portion of the source code corresponding to the message is located using the line number.

229. (original) The computer-readable medium of claim 222, wherein the message comprises a name of a file containing the source code.

230. (original) The computer-readable medium of claim 222, wherein the graphical representation comprises a class diagram.

231. (currently amended) A computer-readable medium containing instructions for controlling a data processing system to perform a method, the data processing system having source code, the method comprising the steps of:

generating a transient meta model which stores a language-neutral representation of the source code;

displaying a graphical representation of the source code generated from the language-neutral representation in the transient meta model;

detecting an error in the source code;

generating a message reflecting the error;

locating a portion of the source code corresponding to the message;

displaying the portion of the source code corresponding to the message; and

displaying the graphical representation of the portion of the source code corresponding to the message in a visually distinctive manner.

232. (original) The computer-readable medium of claim 231, wherein the message is generated by a verification tool.

233. (original) The computer-readable medium of claim 232, wherein the verification tool comprises a compiler.

234. (original) The computer-readable medium of claim 232, wherein the verification tool comprises a quality assurance module.

235. (original) The computer-readable medium of claim 231, wherein the message comprises an error message.

236. (original) The computer-readable medium of claim 231, wherein the message comprises a line number of the source code.

237. (original) The computer-readable medium of claim 236, wherein the portion of the source code corresponding to the message is located using the line number.

238. (original) The computer-readable medium of claim 231, wherein the message comprises a name of a file containing the source code.

239. (original) The computer-readable medium of claim 231, wherein the graphical representation comprises a class diagram.

240. (currently amended) A computer-readable medium containing instructions for controlling a data processing system to perform a method, the data processing system having source code, the method comprising the steps of:

generating a transient meta model which stores a language-neutral representation of the source code;

detecting an error in the source code;

generating a message reflecting the error; and

displaying the graphical representation generated from the language-neutral representation in the transient meta model pertaining to [[of]] a portion of the source code corresponding to the message in a visually distinctive manner.

241. (original) The computer-readable medium of claim 240, wherein the message is generated by a verification tool.

242. (original) The computer-readable medium of claim 241, wherein the verification tool comprises a compiler.

243. (original) The computer-readable medium of claim 241, wherein the verification tool comprises a quality assurance module.

244. (original) The computer-readable medium of claim 240, wherein the message comprises an error message.

245. (original) The computer-readable medium of claim 240, wherein the message comprises a line number of the source code.

246. (original) The computer-readable medium of claim 245, wherein the portion of the source code corresponding to the message is located using the line number.

247. (original) The computer-readable medium of claim 240, wherein the message comprises a name of a file containing the source code.

248. (original) The computer-readable medium of claim 240, wherein the graphical representation comprises a class diagram.

249. (currently amended) A data processing system comprising:

a secondary storage device further comprising source code;

a memory device further comprising a program that generates a transient meta model which stores a language-neutral representation of the source code, that displays a graphical representation of the source code generated from the language-neutral representation in the transient meta model, that receives a message corresponding to a portion of the source code, that locates the portion of the source code corresponding to the message, that displays the portion of the source code corresponding to the message, and that displays the graphical representation of the portion of the source code corresponding to the message in a visually distinctive manner; and

a processor for running the program.

250. (original) The data processing system of claim 249, wherein the program further detects an error in the source code, and generates a message reflecting the error.

251. (original) The data processing system of claim 249, wherein the message is received from a verification tool.

252. (original) The data processing system of claim 251, wherein the verification tool comprises a compiler.

253. (original) The data processing system of claim 251, wherein the verification tool comprises a quality assurance module.

254. (original) The data processing system of claim 249, wherein the message comprises an error message.

255. (original) The data processing system of claim 249, wherein the message comprises a line number of the source code.

256. (original) The data processing system of claim 255, wherein the portion of the source code corresponding to the message is located using the line number.

257. (original) The data processing system of claim 249, wherein the message comprises a name of a file containing the source code.

258. (original) The data processing system of claim 249, wherein the graphical representation comprises a class diagram.

259. (currently amended) A data processing system comprising:

a secondary storage device further comprising source code;

a memory device further comprising a program that generates a transient meta model which stores a language-neutral representation of the source code, that displays a graphical representation of the source code generated from the language-neutral representation in the transient meta model, that detects an error in the source code,

that generates a message reflecting the error, that locates a portion of the source code corresponding to the message, that displays the portion of the source code corresponding to the message, and that displays the graphical representation of the portion of the source code corresponding to the message in a visually distinctive manner; and

a processor for running the program.

260. (original) The data processing system of claim 259, wherein the message is generated by a verification tool.

261. (original) The data processing system of claim 260, wherein the verification tool comprises a compiler.

262. (original) The data processing system of claim 260, wherein the verification tool comprises a quality assurance module.

263. (original) The data processing system of claim 259, wherein the message comprises an error message.

264. (original) The data processing system of claim 259, wherein the message comprises a line number of the source code.

265. (original) The data processing system of claim 264, wherein the portion of the source code corresponding to the message is located using the line number.

266. (original) The data processing system of claim 259, wherein the message comprises a name of a file containing the source code.

267. (original) The data processing system of claim 259, wherein the graphical representation comprises a class diagram.

268. (currently amended) A system for developing source code comprising:

means for generating a transient meta model which stores a language-neutral representation of the source code;

means for receiving a message corresponding to a portion of the source code; and

means for displaying the graphical representation generated from the language-neutral representation in the transient meta model pertaining to [[of]] the portion of the source code corresponding to the message in a visually distinctive manner.